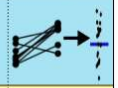
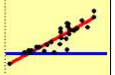
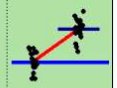
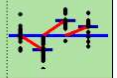
Common statistical tests are linear models (or: how to teach stats)

Jonas Kristoffer Lindeløv

This document is summarised in the table below. It shows the linear models underlying common parametric and non-parametric tests. Formulating all the tests in the same language highlights the many similarities between them. Get it as an image (linear_tests_cheat_sheet.png) or as a PDF (linear_tests_cheat_sheet.pdf).

Common tests are linear models

See worked examples and more details at the accompanying notebook: https://lindeloev.github.io/tests-as-linear

		Built-in function in R	Equivalent linear model in R	Exact?	The linear model in words	Icon
Simple regression: lm(y ~ 1 + x)	**y is independent of x** P: One-sample t-test N: Wilcoxon signed-rank	t.test(y) wilcox.test(y)	lm(y ~ 1) lm(signed_rank(y) ~ 1)	✓ for N >14	One number (intercept, i.e., the mean) predicts **y**. - (Same, but it predicts the *signed rank* of **y**.)	
	P: Paired-sample t-test N: Wilcoxon matched pairs	t.test(y_1, y_2, paired=TRUE) wilcox.test(y_1, y_2, paired=TRUE)	lm(y_2 - y_1 ~ 1) lm(signed_rank(y_2 - y_1) ~ 1)	✓ for N >14	One intercept predicts the pairwise $\mathbf{y_2}$-$\mathbf{y_1}$ differences. - (Same, but it predicts the *signed rank* of $\mathbf{y_2}$-$\mathbf{y_1}$.)	
	y ~ continuous x P: Pearson correlation N: Spearman correlation	cor.test(x, y, method='Pearson') cor.test(x, y, method='Spearman')	lm(y ~ 1 + x) lm(rank(y) ~ 1 + rank(x))	✓ for N >10	One intercept plus **x** multiplied by a number (slope) predicts **y**. - (Same, but with *ranked* **x** and **y**)	
	y ~ discrete x P: Two-sample t-test P: Welch's t-test N: Mann-Whitney U	t.test(y_1, y_2, var.equal=TRUE) t.test(y_1, y_2, var.equal=FALSE) wilcox.test(y_1, y_2)	lm(y ~ 1 + G_2)[A] gls(y ~ 1 + G_2, weights=…[B])[A] lm(signed_rank(y) ~ 1 + G_2)[A]	✓ ✓ for N >11	An intercept for **group 1** (plus a difference if **group 2**) predicts **y**. - (Same, but with one variance *per group* instead of one common.) - (Same, but it predicts the *signed rank* of **y**.)	
Multiple regression: lm(y ~ 1 + x_1 + x_2 +…)	P: One-way ANOVA N: Kruskall-Wallis	aov(y ~ group) kruskal.test(y ~ group)	lm(y ~ 1 + G_2 + G_3 +…+ G_N)[A] lm(rank(y) ~ 1 + G_2 + G_3 +…+ G_N)[A]	✓ for N >11	An intercept for **group 1** (plus a difference if group ≠ 1) predicts **y**. - (Same, but it predicts the *signed rank* of **y**.)	
	P: One-way ANCOVA	aov(y ~ group + x)	lm(y ~ 1 + G_2 + G_3 +…+ G_N + x)[A]	✓	- (Same, but plus a slope on **x**.) *Note: this is discrete AND continuous. ANCOVAs are ANOVAs with a continuous x.*	
	P: Two-way ANOVA	aov(y ~ group * sex)	lm(y ~ 1 + G_2 + G_3 + … + G_N + S_2 + S_3+ … + S_K + G_2*S_2+G_3*S_3+…+G_N*S_K)	✓	Interaction term: changing **sex** changes the **y ~ group** parameters. *Note: $\mathbf{G_{2\,to\,N}}$ is an indicator (0 or 1) for each non-intercept levels of the **group** variable. Similarly for $\mathbf{S_{2\,to\,K}}$ for sex. The first line (with G_i) is main effect of group, the second (with S_i) for sex and the third is the **group × sex** interaction. For two levels (e.g. male/female), line 2 would just be "S_2" and line 3 would be S_2 multiplied with each G_i.*	*[Coming]*
	Counts ~ discrete x N: Chi-square test	chisq.test(groupXsex_table)	**Equivalent log-linear model** glm(y ~ 1 + G_2 + G_3 + … + G_N + S_2 + S_3+ … + S_K + G_2*S_2+G_3*S_3+…+G_N*S_K, family=…)[A]	✓	Interaction term: (Same as Two-way ANOVA.) *Run glm using the following arguments:* `glm(model, family=poisson())` *As linear-model, the Chi-square test is $\log(y_i) = \log(N) + \log(\alpha_i) + \log(\beta_j) + \log(\alpha_i\beta_j)$ where α_i and β_j are proportions. See more info in the accompanying notebook.*	*Same as Two-way ANOVA*
	N: Goodness of fit	chisq.test(y)	glm(y ~ 1 + G_2 + G_3 +…+ G_N, …)[A]	✓	(Same as One-way ANOVA.)	*1W-ANOVA*

List of parametric (P) non-parametric (N) tests and equivalent linear models. The notation y ~ 1 + x is R shorthand for $y = 1 \cdot b + a \cdot x$ which most of us learned in school. Models in similar colors are highly similar, but really, notice how similar they all are! For non-parametric models, the linear models are reasonable approximations for non-small sample sizes (see "Exact" column and click links to see simulations). Other less accurate approximations exist, e.g., Wilcoxon for the sign test and Goodness-of-fit for the binomial test. The signed rank function is `signed_rank = function(x) sign(x) * rank(abs(x))`. The variables G_i and S_i are "dummy coded" indocator variables (either 0 or 1) exploiting the fact that when $\Delta x = 1$ between categories the difference equals the slope. Subscripts (e.g., G_2 or y_1) indicate different columns in data. lm requires long-format data for all non-continuous models. All of this is exposed in greater detail and worked examples at https://lindeloev.github.io/tests-as-linear.

[A] See the note to the two-way ANOVA for explanation of the notation.
[B] Same model, but with one variance per group: `gls(value ~ 1 + G2, weights = varIdent(form = ~1|group), method="ML")`.

Jonas Kristoffer Lindeløv
http://lindeloev.net

(linear_tests_cheat_sheet.pdf)

1 The simplicity underlying common tests

Most of the common statistical models (t-test, correlation, ANOVA; chi-square, etc.) are special cases of linear models or a very close approximation. This beautiful simplicity means that there is less to learn. In particular, it all comes down to $y = a \cdot x + b$ which most students know from highschool. Unfortunately, stats intro courses are usually taught as if each test is an independent tool, needlessly making life more complicated for students and teachers alike.

This needless complexity multiplies when students try to rote learn the parametric assumptions underlying each test separately rather than deducing them from the linear model.

For this reason, I think that teaching linear models first and foremost and *then* name-dropping the special cases along the way makes for an excellent teaching strategy, emphasizing *understanding* over rote learning. Since linear models are the same across frequentist, Bayesian, and permutation-based inferences, I'd argue that it's better to start with modeling than p-values, type-1 errors, Bayes factors, or other inferences.

Concerning the teaching of *non-parametric* tests in intro-courses, I think that we can justify lying-to-children (https://en.wikipedia.org/wiki/Lie-to-children) and teach non-parametric tests as if they are merely ranked versions of the corresponding parametric tests. It is much better for students to think "ranks!" than to believe that you can magically throw away assumptions. Indeed, the Bayesian equivalents of non-parametric tests implemented in JASP (https://jasp-stats.org) literally just do (latent) ranking (https://arxiv.org/abs/1712.06941) and that's it. For the frequentist non-parametric tests considered here, this approach is highly accurate for N > 15.

Use the menu to jump to your favourite section. There are links to lots of similar (though more scattered) stuff under sources and teaching materials. I hope that you will join in suggesting improvements or submitting improvements yourself in the Github repo to this page (https://github.com/lindeloev/tests-as-linear). Let's make it awesome!

2 Settings and toy data

Unfold this if you want to see functions and other settings for this notebook:

Show Source

For a start, we'll keep it simple and play with three standard normals in wide (`a` , `b` , `c`) and long format (`value` , `group`):

```
# Wide format (sort of)
y = rnorm_fixed(50, mu=0.3, sd=2)  # Almost zero mean
x = rnorm_fixed(50, mu=0, sd=1)  # Used in correlation where this is on x-axis
y2 = rnorm_fixed(50, mu=0.5, sd=1.5)  # Used in two means

# Long format data with indicator
value = c(y, y2)
group = rep(c('y1', 'y2'), each = 50)

# We'll need the signed rank function for a lot of the "non-parametric" tests
signed_rank = function(x) sign(x) * rank(abs(x))
```

3 Pearson and Spearman correlation

3.0.1 Theory: As linear models

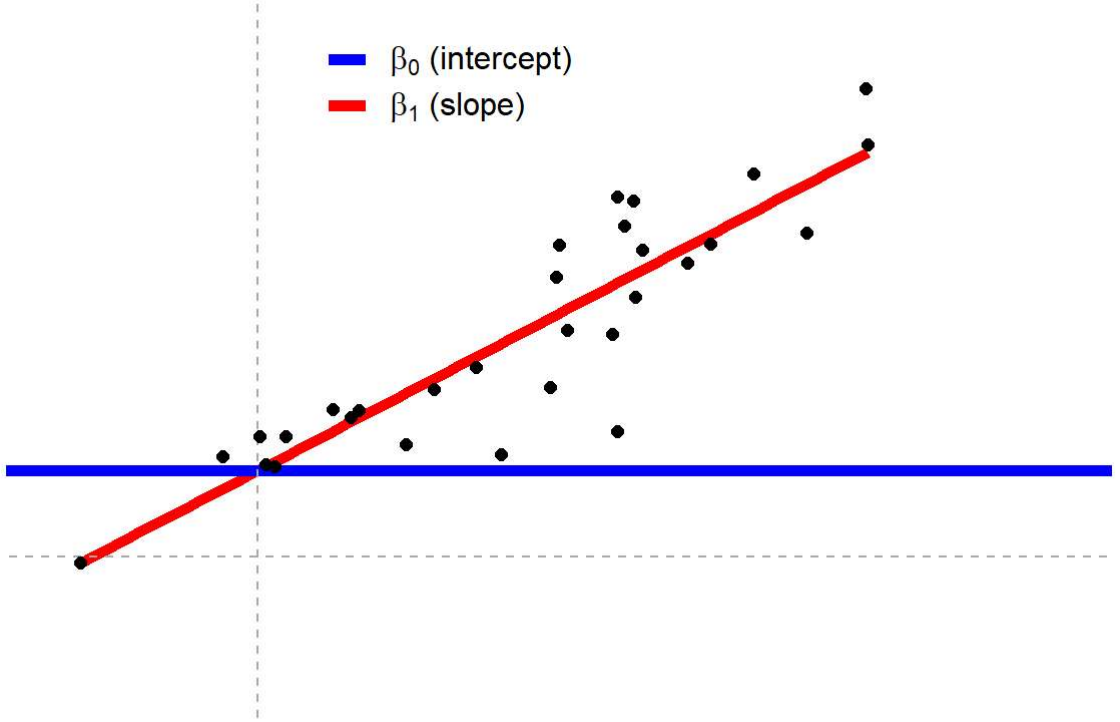
Model: the recipe for y is a slope (β_1) times x plus an intercept (β_0, aka a straight line).

$$y = \beta_0 + \beta_1 x \qquad \mathcal{H}_0 : \beta_1 = 0$$

… which is a math-y way of writing the good old $y = ax + b$ (here ordered as $y = b + ax$). In R we are lazy and write `y ~ 1 + x` which R reads like `y = 1*number + x*othernumber` and the task of t-tests, lm, etc., is simply to find the numbers that best predict y.

Either way you write it, it's an intercept (β_0) and a slope (β_1) yielding a straight line:

Show Source



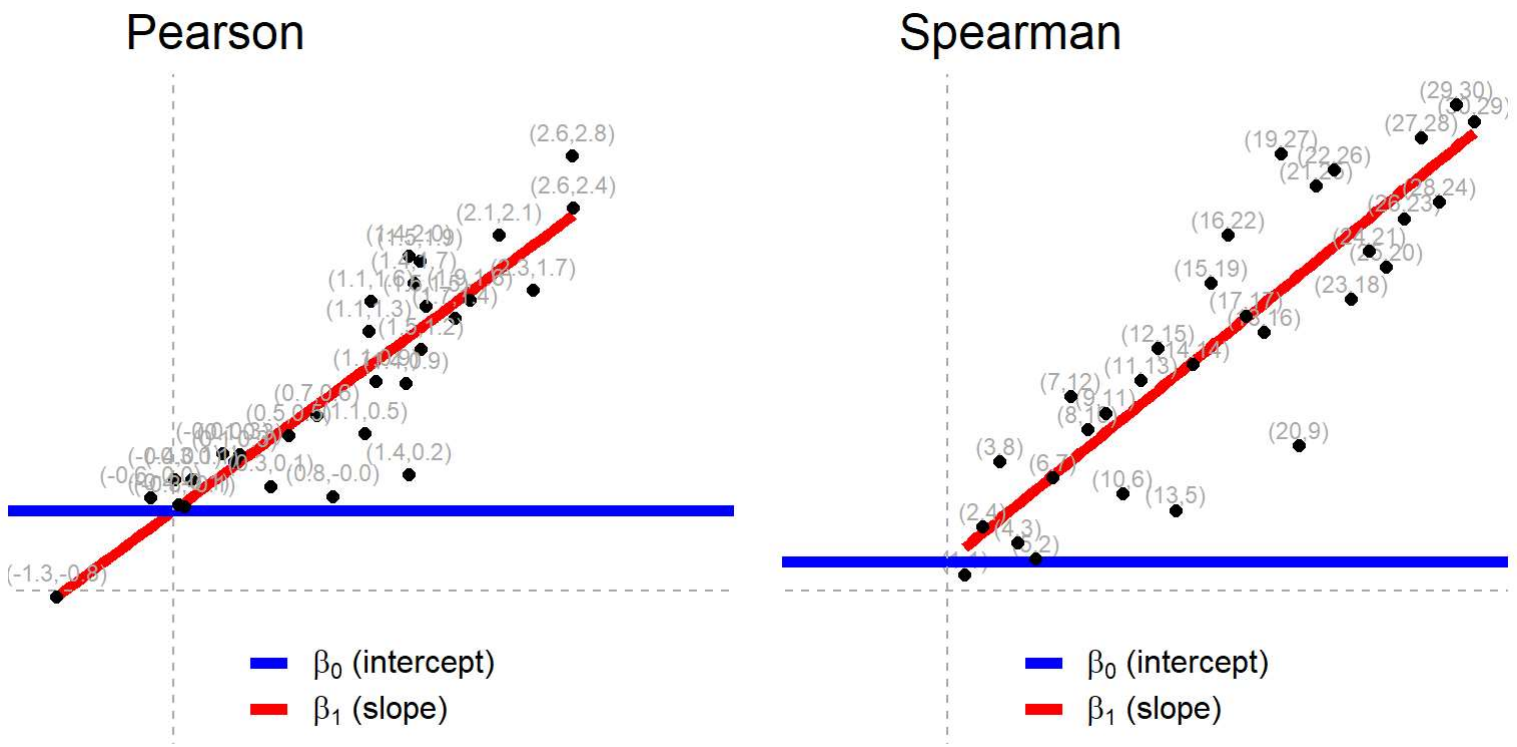
This is often simply called a **regression** model which can be extended to **multiple regression** where there are several βs and on the right-hand side multiplied with the predictors. Everything below, from one-sample t-test to two-way ANOVA are just special cases of this system. Nothing more, nothing less.

As the name implies, the **Spearman rank correlation** is a **Pearson correlation** on rank-transformed x and y:

$$rank(y) = \beta_0 + \beta_1 \cdot rank(x) \qquad \mathcal{H}_0 : \beta_1 = 0$$

The correlation coefficient of the linear model is identical to a “real” Pearson correlation, but p-values are an approximation which is is appropriate for samples greater than N=10 and almost perfect when N > 20 (simulate_spearman.html). Such a nice and non-mysterious equivalence that many students are left unaware of! Visualizing them side by side including data labels, we see this rank-transformation in action:

Show Source



3.0.2 R code: Pearson correlation

It couldn't be much simpler to run these models in R. They yield identical `p` and `t`, but there's a catch: `lm` gives you the *slope* and even though that is usually much more interpretable and informative than the *correlation coefficient r*, you may still want *r*. Luckily, the slope becomes `r` if `x` and `y` have a standard deviation of exactly 1. You can do this using `scale(x)` or `I(x/sd(x))`:

```
a = cor.test(y, x, method = "pearson") # Built-in
b = lm(y ~ 1 + x) # Equivalent linear model: y = Beta0*1 + Beta1*x
c = lm(scale(y) ~ 1 + scale(x))  # On scaled vars to recover r
```

Results:

model	r	p.value	t	conf.low	conf.high

model	r	p.value	t	conf.low	conf.high
cor.test	-0.2318	0.1053	-1.6507	-0.4792	0.0498
lm	-0.4636	0.1053	-1.6507	-1.0282	0.1011
lm scaled	-0.2318	0.1053	-1.6507	-0.5141	0.0505

Show R output

The CIs are not exactly identical, but very close.

3.0.3 R code: Spearman correlation

Note that we can interpret the slope which is the number of ranks y change for each rank on x. I think that this is a pretty interesting number. However, the intercept is less interpretable since it lies at $rank(x) = 0$ which is impossible since x starts at 1.

See the identical `r` (now "rho") and `p`:

```
# Spearman correlation
a = cor.test(y, x, method = "spearman") # Built-in
b = lm(rank(y) ~ 1 + rank(x)) # Equivalent linear model
```

Let's look at the results:

model	p.value	rho
cor.test	0.1135	-0.2266
lm	0.1135	-0.2266

Show R output

4 One mean

4.1 One sample t-test and Wilcoxon signed-rank

4.1.1 Theory: As linear models

t-test model: A single number predicts y.

$y = \beta_0 \qquad \mathcal{H}_0 : \beta_0 = 0$

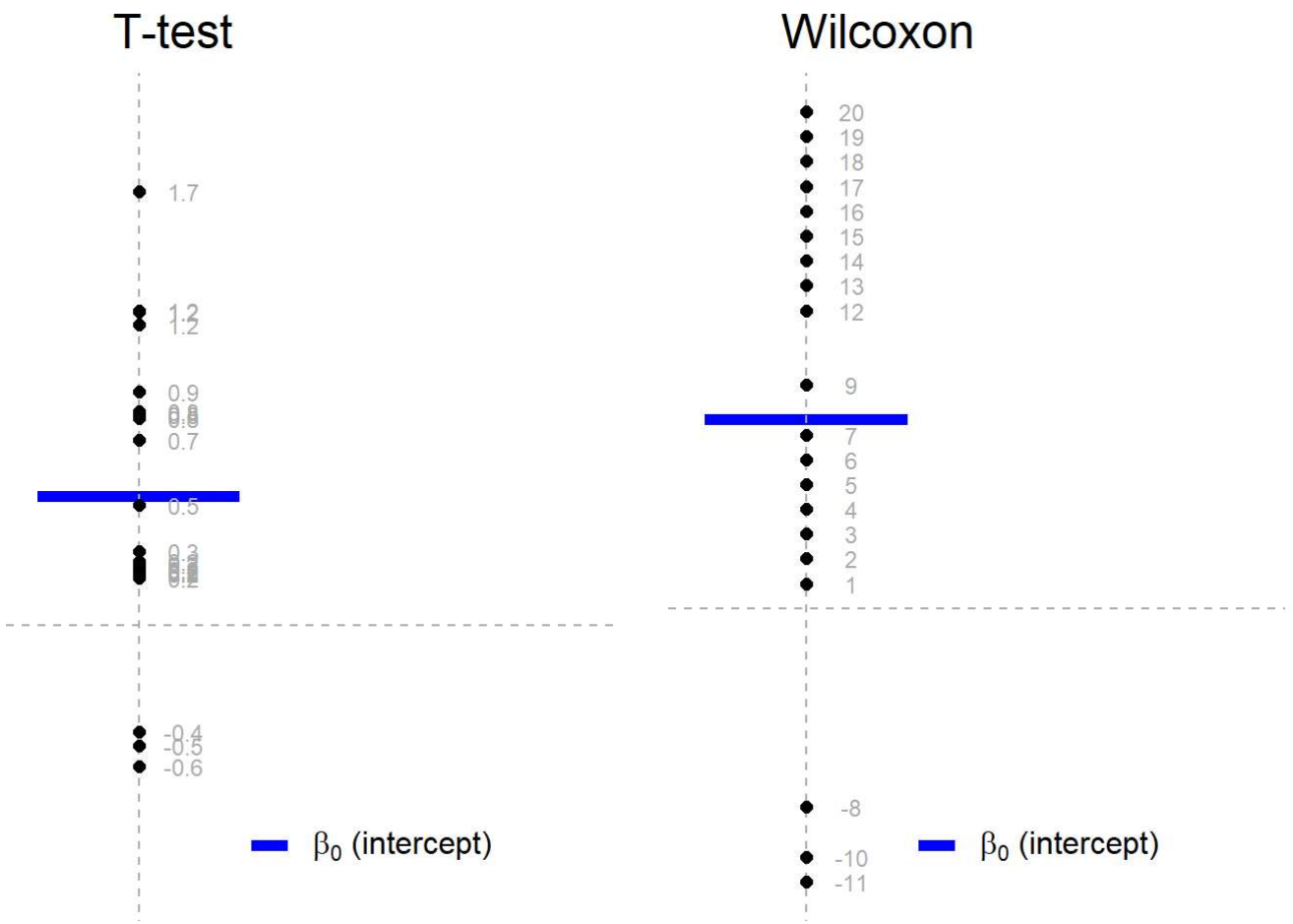
In other words, it's our good old $y = \beta_0 + \beta_1 * x$ where the last term is gone since there is no x (essentially $x = 0$, see left figure below).

The same is to a very close approximately true for **Wilcoxon signed-rank test**, just with the signed ranks of y instead of y itself (see right panel below and caveat in the end of this section):

$signed\_rank(y) = \beta_0$

This approximation is good enough when the sample size is larger than 14 and almost perfect if the sample size is larger than 50 (simulate_wilcoxon.html).

Show Source



One interesting implication is that *many "non-parametric tests" are precisely as parametric as their parametric counterparts with means, standard deviations, homogeneity of variance, etc. - just on transformed data*.

4.1.2 R code: One-sample t-test

Try running the R code below and see that the linear model (`lm`) produces the same t, p, and r as the built-in `t.test` . The confidence interval is not presented in the output of `lm` but is also identical if you use `confint(lm(...))` :

```
# Built-in t-test
a = t.test(y)

# Equivalent linear model: intercept-only
b = lm(y ~ 1)
```

Results:

model	mean	p.value	t	df	conf.low	conf.high
t.test	0.3	0.294	1.0607	49	-0.2684	0.8684
lm	0.3	0.294	1.0607	49	-0.2684	0.8684

Show R output

4.1.3 R code: Wilcoxon signed-rank test

In addition to matching `p`-values, `lm` also gives us the mean signed rank, which I find to be an informative number.

```
# Built-in
a = wilcox.test(y)

# Equivalent linear model
b = lm(signed_rank(y) ~ 1)  # See? Same as above, just on signed ranks

# Bonus: of course also works for one-sample t-test
c = t.test(signed_rank(y))
```

Results:

model	p.value	mean_rank

model	p.value	mean_rank
wilcox.test	0.3693	
lm	0.3721	3.74
t.test	0.3721	3.74

Show R output

4.2 Paired samples t-test and Wilcoxon matched pairs

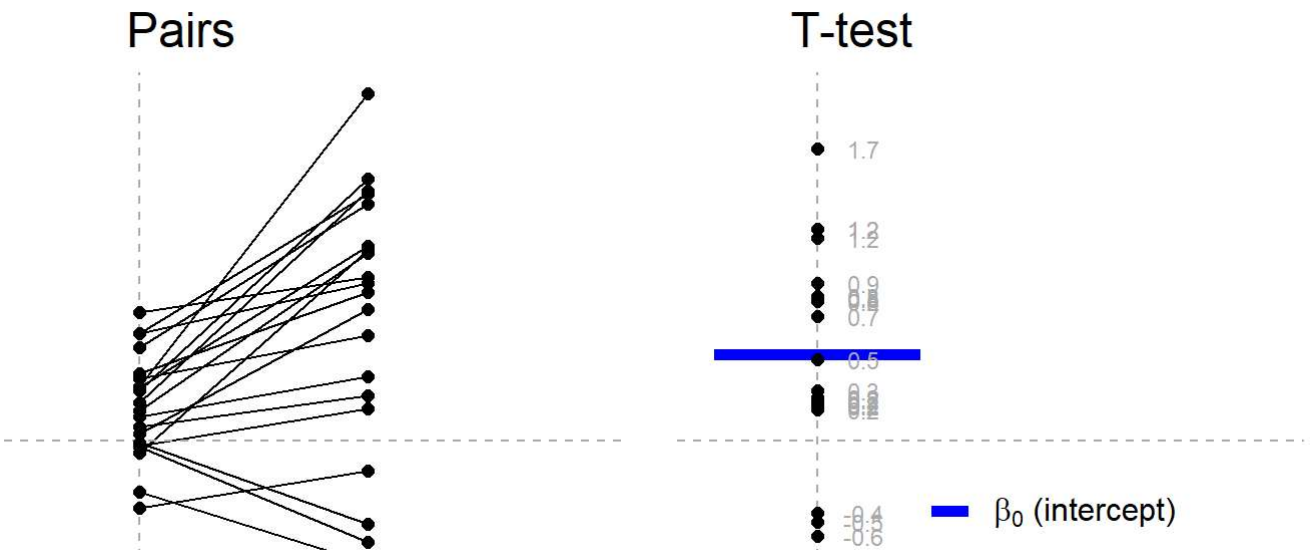
4.2.1 Theory: As linear models

t-test model: a single number (intercept) predicts the pairwise differences.

$y_2 - y_1 = \beta_0 \qquad \mathcal{H}_0 : \beta_0 = 0$

This means that there is just one $y = y_2 - y_1$ to predict and it becomes a one-sample t-test on the pairwise differences. The visualization is therefore also the same as for the one-sample t-test. At the risk of overcomplicating a simple substraction, you can think of these pairwise differences as slopes (see left panel of the figure), which we can represent as y-offsets (see right panel of the figure):

Show Source



Similarly, the **Wilcoxon matched pairs** only differ from **Wilcoxon signed-rank** in that it's testing the signed ranks of the pairwise $y - x$ differences.

$$signed\_rank(y_2 - y_1) = \beta_0 \qquad \mathcal{H}_0 : \beta_0 = 0$$

4.2.2 R code: Paired sample t-test

```
a = t.test(y, y2, paired = TRUE) # Built-in paired t-test
b = lm(y - y2 ~ 1) # Equivalent linear model
```

Results:

model	mean	p.value	df	t	conf.low	conf.high
t.test	-0.2	0.601	49	-0.5264	-0.9635	0.5635
lm	-0.2	0.601	49	-0.5264	-0.9635	0.5635

Show R output

4.2.3 R code: Wilcoxon matched pairs

Again, we do the signed-ranks trick. This is still an approximation, but a close one:

```
# Built-in Wilcoxon matched pairs
a = wilcox.test(y, y2, paired = TRUE)

# Equivalent linear model:
b = lm(signed_rank(y - y2) ~ 1)

# Bonus: identical to one-sample t-test ong signed ranks
c = t.test(signed_rank(y - y2))
```

Results:

model	p.value	mean_rank_diff
wilcox.test	0.8243	
lm	0.8232	-0.94
t.test	0.8232	-0.94

Show R output

For large sample sizes (N >> 100), this approaches the **sign test** to a reasonable degree, but this approximation is too inaccurate to flesh out here.

5 Two means

5.1 Independent t-test and Mann-Whitney U

5.1.1 Theory: As linear models

Independent t-test model: two means predict y.

$$y_i = \beta_0 + \beta_1 x_i \qquad \mathcal{H}_0 : \beta_1 = 0$$

where x_i is an indicator (0 or 1) saying whether data point i was sampled from one or the other group. Indicator variables (also called “dummy coding”) (https://en.wikipedia.org/wiki/Dummy_variable_(statistics)) underly a lot of linear models and we’ll take an aside to see how it works in a minute.

Mann-Whitney U (also known as **Wilcoxon signed-rank test** for two independent groups) is the same model to a very close approximation, just on the ranks of x and y instead of the actual values:

$$rank(y_i) = \beta_0 + \beta_1 x_i \qquad \mathcal{H}_0 : \beta_1 = 0$$

To me, equivalences like this make “non-parametric” statistics much easier to understand. The approximation is appropriate when the sample size is larger than 11 in each group and virtually perfect when N > 30 in each group (simulate_mannwhitney.html).

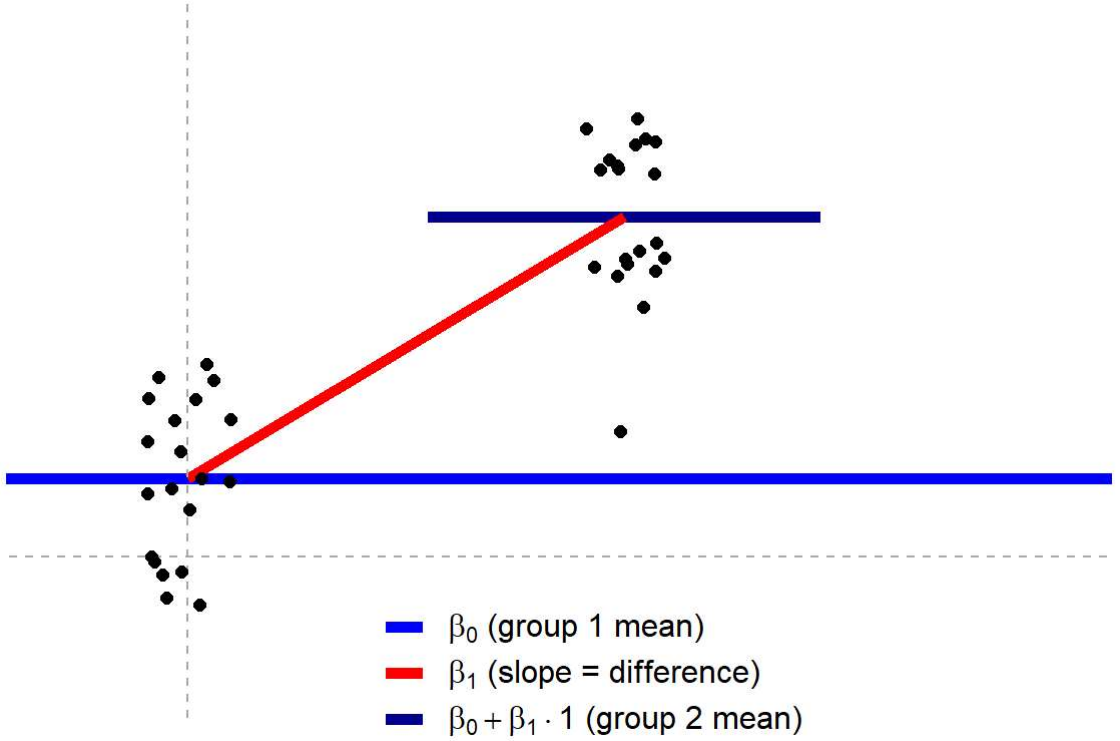
5.1.2 Theory: Dummy coding

Dummy coding can be understood visually. The indicator is on the x-axis so data points from the first group are located at $x = 0$ and data points from the second group is located at $x = 1$. Then β_0 is the intercept (red line) and β_1 is the slope between the two means (green line). Why? Because when $\Delta x = 1$ the slope equals the difference because:

$$slope = \Delta y / \Delta x = \Delta y / 1 = \Delta y = difference$$

Magic! Even categorical differences can be modelled using linear models! It’s a true Swizz army knife.

Show Source



5.1.3 Theory: Dummy coding (continued)

If you feel like you get dummy coding now, just skip ahead to the next section. Here is a more elaborate explanation of dummy coding:

If a data point was sampled from the first group, i.e., when $x_i = 0$, the model simply becomes $y = \beta_0 + \beta_1 \cdot 0 = \beta_0$. In other words, the model predicts that that data point is $beta_0$. It turns out that the β which best predicts a set of data points is the *mean* of those data points, so β_0 is the mean of group 1.

On the other hand, data points sampled from the second group would have $x_i = 1$ so the model becomes $y_i = \beta_0 + \beta_1 \cdot 1 = \beta_0 + \beta_1$. In other words, we add β_1 to "shift" from the mean of the first group to the mean of the second group. Thus β_1 becomes the *mean difference* between the groups.

As an example, say group 1 is 25 years old ($\beta_0 = 25$) and group 2 is 28 years old ($\beta_1 = 3$), then the model for a person in group 1 is $y = 25 + 3 \cdot 0 = 25$ and the model for a person in group 2 is $y = 25 + 3 \cdot 1 = 28$.

Hooray, it works! For first-timers it takes a few moments to understand dummy coding, but you only need to know addition and multiplication to get there!

5.1.4 R code: independent t-test

As a reminder, when we write `y ~ 1 + x` in R, it is shorthand for $y = \beta_0 \cdot 1 + \beta_1 \cdot x$ and R goes on computing the βs for you. Thus $y\ 1 + x$ is the R-way of writing $y = a \cdot x + b$.

Notice the identical `t`, `df`, `p`, and estimates. We can get the confidence interval by running `confint(lm(...))`.

```
# Built-in independent t-test on wide data
a = t.test(y, y2, var.equal = TRUE)

# Be explicit about the underlying linear model by hand-dummy-coding:
group_y2 = ifelse(group == 'y2', 1, 0)  # 1 if group == y2, 0 otherwise
b = lm(value ~ 1 + group_y2)  # Using our hand-made dummy regressor

# Note: We could also do the dummy-coding in the model
# specification itself. Same result.
c = lm(value ~ 1 + I(group=='y2'))
```

Results:

model	mean_y	difference	p.value	df	conf.low	conf.high
t.test	0.3	0.2	0.5729	98	-0.5016	0.9016
lm	0.3	0.2	0.5729	98	-0.5016	0.9016

Show R output

5.1.5 R code: Mann-Whitney U

```
# Wilcoxon / Mann-Whitney U
a = wilcox.test(y, y2)

# As linear model with our dummy-coded group_y2:
b = lm(rank(value) ~ 1 + group_y2)  # compare to linear model above
```

model	p.value	rank_diff
wilcox.test	0.7907	
lm	0.7896	1.56

Show R output

5.2 Welch's t-test

This is identical to the (Student's) independent t-test above except that Student's assumes identical variances and **Welch's t-test** does not. So the linear model is the same and the trick is in the variances, which I won't go further into here.

```
# Built-in
a = t.test(y, y2, var.equal=FALSE)

# As linear model with per-group variances
b = nlme::gls(value ~ 1 + group_y2, weights = nlme::varIdent(form=~1|group), method="ML")
```

Results:

model	mean_y	mean_diff	p.value	t	conf.low	conf.high
t.test	0.3	0.2	0.573	-0.5657	-0.9023	0.5023

model	mean_y	mean_diff	p.value	t	conf.low	conf.high
gls	0.3	0.2	0.5729	-0.5657	-0.893	0.493

Show R output

6 Three or more means

ANOVAs are linear models with (only) categorical predictors so they simply extend everything we did above, relying heavily on dummy coding. Do make sure to read the section on dummy coding if you haven't already.

6.1 One-way ANOVA and Kruskal-Wallis

6.1.1 Theory: As linear models

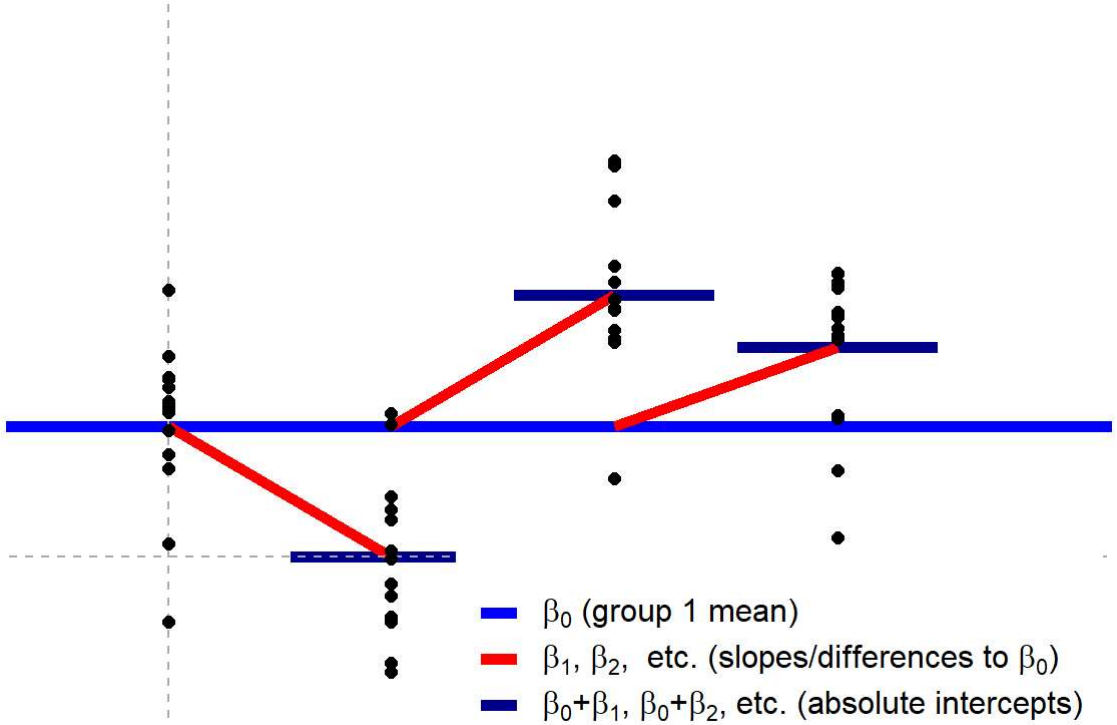
Model: One mean for each group predicts y.

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \dots \qquad \mathcal{H}_0 : y = \beta_1$

where x_i are indicators ($x = 0$ or $x = 1$) where at most one $x_i = 1$ while all others are $x_i = 0$.

Notice how this is just "more of the same" of what we already did in other models above. When there are only two groups, this model is $y = \beta_0 + \beta_1 * x$, i.e. the independent t-test. If there is only one group, it is $y = \beta_0$, i.e. the one-sample t-test. This is easy to see in the visualization below - just cover up a few groups and see that it matches the other visualizations above, though I did omit adding green lines from the intercept (red) to the group means (blue) for visual clarity.

Show Source



A one-way ANOVA has a log-linear counterpart called goodness-of-fit test which we'll return to. By the way, since we now regress on more than one x, the one-way ANOVA is a **multiple regression** model.

The **Kruskal-Wallis** test is simply a **one-way ANOVA** on the rank-transformed y (`value`):

$$rank(y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots$$

This approximation is good enough for 12 or more data points (simulate_kruskall.html). Again, if you do this for just one or two groups, we're already acquainted with those equations, i.e. the Wilcoxon signed-rank test or the Mann-Whitney U test respectively.

6.1.2 Example data

We make a three-level factor with the levels `a` , `b` , and `c` so that the **one-way ANOVA** basically becomes a "three-sample t-test". Then we manually do the dummy coding of the groups.

```
# Three variables in "long" format
N = 20  # Number of samples per group
D = data.frame(
   value = c(rnorm_fixed(N, 0), rnorm_fixed(N, 1), rnorm_fixed(N, 0.5)),
   group = rep(c('a', 'b', 'c'), each=N),

   # Explicitly add indicator/dummy variables
   # Could also be done using model.matrix(~D$group)
   group_a = rep(c(1, 0, 0), each=N),
   group_b = rep(c(0, 1, 0), each=N),
   group_c = rep(c(0, 0, 1), each=N))  # N of each level
```

value	group	group_a	group_b	group_c
1.2054	a	1	0	0
-0.3238	a	1	0	0
-2.2574	a	1	0	0
-0.3088	a	1	0	0
0.8638	a	1	0	0
0.9145	a	1	0	0
1.2604	a	1	0	0
-0.4439	a	1	0	0
-0.6537	a	1	0	0
-0.6647	a	1	0	0

Showing 1 to 10 of 60 entries Previous 1 2 3 4 5 6 Next

See? Exactly one parameter predicts a `value` in a given row while the others are not included in the modeling of that `value`.

6.1.3 R code: one-way ANOVA

OK, let's see the identity between the built-in **ANOVA** (`car::Anova`) and the dummy-coded in-your-face linear model in `lm`. Actually, `car::Anova` and `aov` are just wrappers around `lm` so the identity comes as no surprise. The latter returns parameter estimates as well (bonus!), but we'll just look at the overall model statistics for now. Note that I do not use the `aov` function because it computes type-I sum of squares. There is a BIG polarized debate about whether to use type-II (as `car::Anova` does by default) or type-III sum of squares, but let's skip that for now.

```
# Compare built-in and linear model
a = car::Anova(aov(value ~ group, D))  # Built-in ANOVA
b = lm(value ~ 1 + group_a + group_b + group_c, data=D)  # As in-your-face linear model
```

Results:

model	F	p.value	df	df.residual
Anova	5	0.01	2	57
lm	5	0.01	3	57

Show R output

6.1.4 R code: Kruskal-Wallis

```
a = kruskal.test(value ~ group, D)  # Built-in
b = lm(rank(value) ~ 1 + group_a + group_b + group_c, D)  # As linear model
c = car::Anova(aov(rank(value) ~ group, D))  # Of course the same using the built-in ANOVA, which is just a wrapper arou
nd lm
```

Results:

model	p.value
kruskal.test	0.0355
lm	0.0326

Show R output

6.2 Two-way ANOVA (plot in progress!)

6.2.1 Theory: As linear models

Model: one mean per group (main effects) plus these means multiplied across factors (interaction effects). The main effects are the one-way ANOVAs above, though in the context of a larger model. The interaction effect is harder to explain in the abstract even though it's just a few numbers multiplied with each other. I will leave that to the teachers to keep focus on equivalences here :-)

Switching to matrix notation:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 \qquad \mathcal{H}_0 : \beta_3 = 0$$

Here β_i are vectors of betas of which only one is selected by the indicator vector X_i. The $\mathcal{H}_0$ shown here is the interaction effect. Note that the intercept β_0, to which all other βs are relative, is now the mean for the first level of all factors.

Continuing with the dataset from the one-way ANOVA above, let's add a crossing factor `mood` so that we can test the `group:mood` interaction (a 3x2 ANOVA). We also do the dummy coding of this factor needed for the linear model.

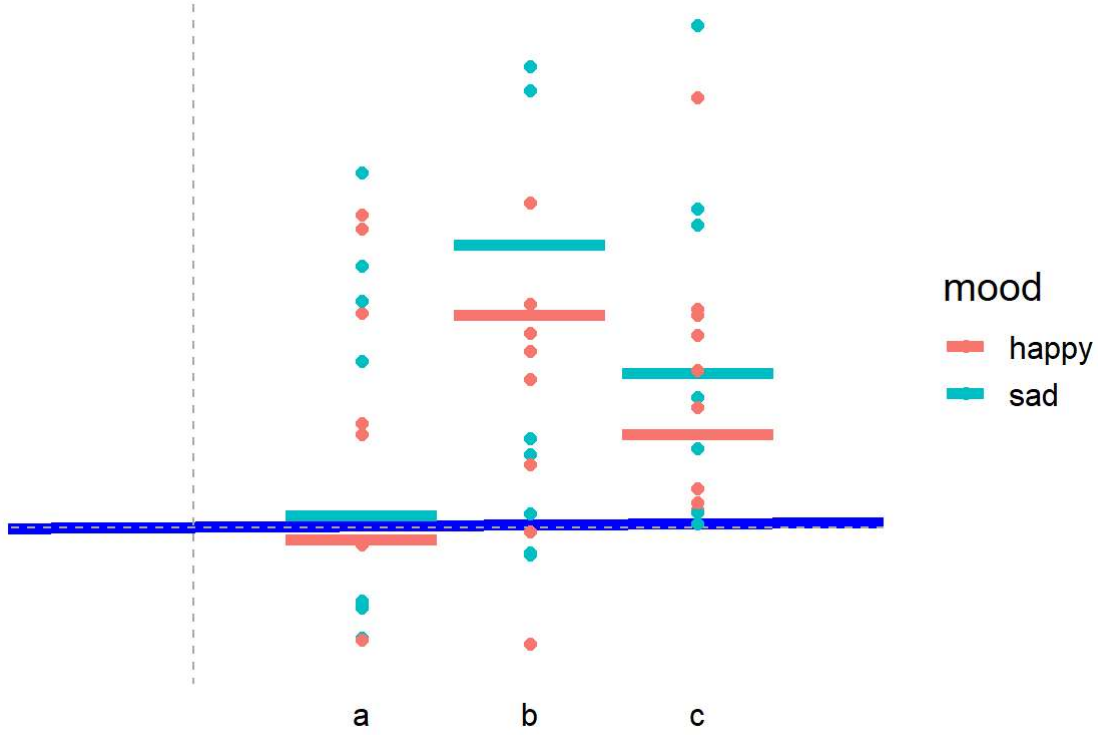
```
# Crossing factor
D$mood = c('happy', 'sad')

# Dummy coding
D$mood_happy = ifelse(D$mood == 'happy', 1, 0)  # 1 if mood==happy. 0 otherwise.
D$mood_sad = ifelse(D$mood == 'sad', 1, 0)  # Same, but we won't be needing this
```

value	group	group_a	group_b	group_c	mood	mood_happy	mood_sad
1.2054	a	1	0	0	happy	1	0
-0.3238	a	1	0	0	sad	0	1
-2.2574	a	1	0	0	happy	1	0
-0.3088	a	1	0	0	sad	0	1
0.8638	a	1	0	0	happy	1	0
0.9145	a	1	0	0	sad	0	1
1.2604	a	1	0	0	happy	1	0
-0.4439	a	1	0	0	sad	0	1
-0.6537	a	1	0	0	happy	1	0
-0.6647	a	1	0	0	sad	0	1

Showing 1 to 10 of 60 entries Previous 1 2 3 4 5 6 Next

β_0 is now the happy guys from group a!

Show Source



6.2.2 R code: Two-way ANOVA

Now let's turn to the actual modeling in R. We compare the built-in ANOVA function to the linear model using `lm`. Notice that in ANOVA, we are testing a full factor interaction all at once which involves many parameters (two in this case), so we can't look at the overall model fit nor any particular parameter for the result. Therefore, I use a likelihood-ratio test to compare a full two-way ANOVA model ("saturated") to one without the interaction effect(s). We do so using the `anova` function. Even though that looks like cheating, it's just computing likelihoods, p-values, etc. on the models that were already fitted, so it's legit!

```
# Built-in two-way ANOVA.
a = car::Anova(aov(value ~ mood * group, D), type='II')  # Normal notation. "*" both multiplies and adds main effects
b = car::Anova(aov(value ~ mood + group + mood:group, D))  # Identical but more verbose about main effects and interacti
on

# Testing the interaction terms as linear model.
full = lm(value ~ 1 + group_a + group_b + mood_happy + group_a:mood_happy + group_b:mood_happy, D)  # Full model
null = lm(value ~ 1 + group_a + group_b + mood_happy, D)  # Without interaction
c = anova(null, full)  # whoop whoop, same F, p, and Dfs
```

Results:

model	F	df	p.value	sumsq	res.sumsq
Anova mood:group	0.0459	2	0.9552	0.0956	56.2468
lm LRT	0.0459	2	0.9552	0.0956	56.2468

Show R output

Below, I present approximate main effect models, though exact calculation of ANOVA main effects is more involved (https://stats.idre.ucla.edu/stata/faq/how-can-i-get-anova-simple-main-effects-with-dummy-coding/) if it is to be accurate and furthermore depend on whether type-II or type-III sum of squares are used for inference.

Look at the model summary statistics to find values comparable to the `Anova` -estimated main effects above.

```
# Main effect of group.
e = lm(value ~ 1 + group_a + group_b, D)

# Main effect of mood.
f = lm(value ~ 1 + mood_happy, D)
```

term	model	F	p.value

term	model	F	p.value
group	Anova	4.8003	0.0121
group	lm	5	0.01
mood	Anova	0.6314	0.4303
mood	lm	0.575	0.4514

Show R output

6.3 ANCOVA

This is simply ANOVA with a continuous regressor added so that it now contains continuous and (dummy-coded) categorical predictors. For example, if we continue with the one-way ANOVA example, we can add `age` and it is now called a **one-way ANCOVA**:

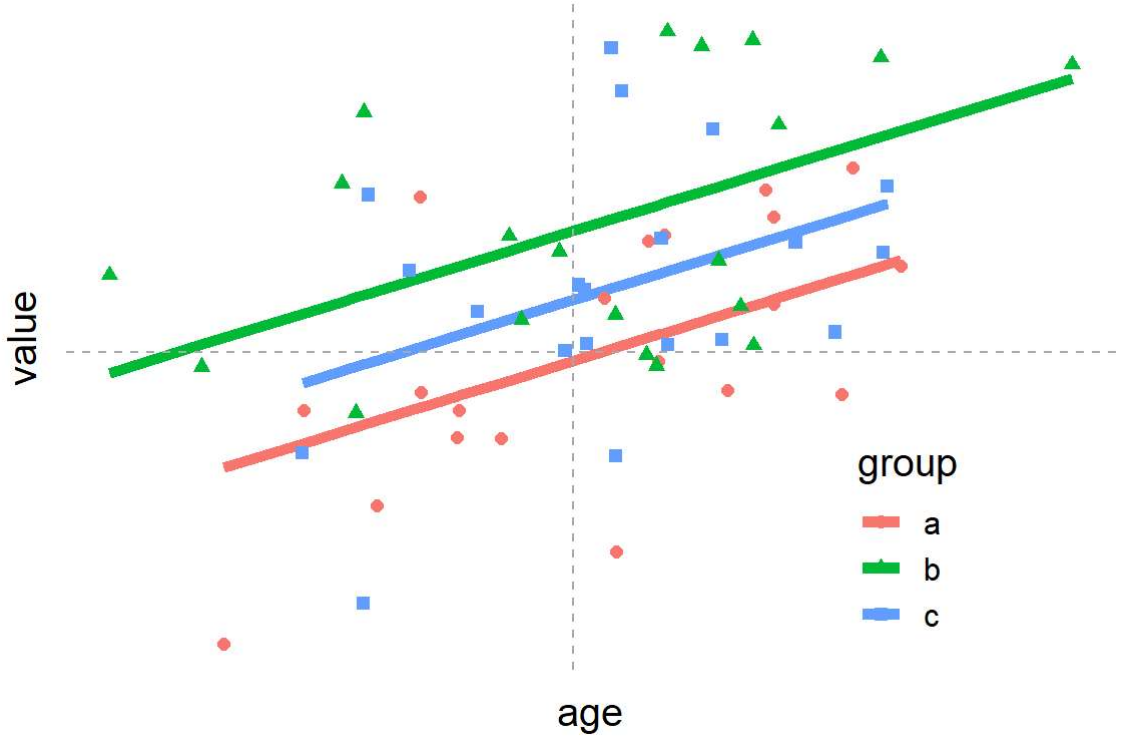
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_3 age$$

… where x_i are our usual dummy-coded indicator variables. β_0 is now the mean for the first group at $age = 0$. You can turn all ANOVAs into ANCOVAs this way, e.g. by adding $\beta_N \cdot age$ to our **two-way ANOVA** in the previous section. But let us go ahead with our one-way ANCOVA, starting by adding age to our dataset:

```
# Update data with a continuous covariate
D$age = D$value + rnorm_fixed(nrow(D), sd=3)  # Correlated to value
```

This is best visualized using colors for groups instead of x-position. The βs are still the average y-offset of the data points, only now we model each group using a slope instead of an intercept. In other words, the one-way ANOVA is sort of one-sample t-tests model for each group ($y = \beta_0$) while the **one-way ANCOVA** is sort of Pearson correlation model for each group ($y_i = \beta_0 + \beta_i + \beta_1 \cdot age$):

Show Source



And now some R code to run the one-way ANCOVA as a linear model:

```
# Built-in ANCOVA. The order of factors matter in pure-aov (type-I variance).
# Use type-II or type-III instead; implemented in car::Anova
a = car::Anova(aov(value ~ group + age, D))
#a = aov(value ~ group + age, D)  # Predictor order matters. Not nice!

# As dummy-coded linear model.
full = lm(value ~ 1 + group_a + group_b + age, D)

# Testing main effect of age using Likelihood-ratio test
null_age = lm(value ~ 1 + group_a + group_b, D)  # Full without age. One-way ANOVA!
result_age = anova(null_age, full)

# Testing main effect of groupusing Likelihood-ratio test
null_group = lm(value ~ 1 + age, D)  # Full without group. Pearson correlation!
result_group = anova(null_group, full)
```

Results:

term	model	F	df	p.value	sumsq	res.sumsq	res.df
age	Anova	16.3967	1	0.0002	12.9096	44.0904	56
age	lm	16.3967	1	0.0002	12.9096	44.0904	56
group	Anova	6.4258	2	0.0031	10.1184	44.0904	56
group	lm	6.4258	2	0.0031	10.1184	44.0904	56

Show R output

7 Proportions: Chi-square is a log-linear model

Recall that when you take the logarithm, you can easily make statements about *proportions*, i.e., that for every increase in x, y increases a certain percentage. This turns out to be one of the simplest (and therefore best!) ways to make count data and contingency tables intelligible. See this nice introduction (https://www.uni-tuebingen.de/fileadmin/Uni_Tuebingen/SFB/SFB_833/A_Bereich/A1/Christoph_Scheepers_-_Statistikworkshop.pdf) to Chi-Square tests as linear models.

7.1 Goodness of fit

7.1.1 Theory: As log-linear model

Model: a single intercept predicts $log(y)$.

I'll refer you to take a look at the section on contingency tables which is basically a "two-way goodness of fit".

7.1.2 Example data

For this, we need some wide count data:

```
# Data in long format
D = data.frame(mood = c('happy', 'sad', 'meh'),
               counts = c(60, 90, 70))

# Dummy coding for the linear model
D$mood_happy = ifelse(D$mood == 'happy', 1, 0)
D$mood_sad = ifelse(D$mood == 'sad', 1, 0)
```

mood	counts	mood_happy	mood_sad
happy	60	1	0
sad	90	0	1
meh	70	0	0

7.1.3 R code: Goodness of fit

Now let's see that the Goodness of fit is just a log-linear equivalent to a one-way ANOVA. We set `family = poisson()` which defaults to setting a logarithmic link function (https://en.wikipedia.org/wiki/Generalized_linear_model#Link_function) (`family = poisson(link='log')`).

```
# Built-in test
a = chisq.test(D$counts)

# As log-linear model, comparing to an intercept-only model
full = glm(counts ~ 1 + mood_happy + mood_sad, data=D, family=poisson())
null = glm(counts ~ 1, data=D, family=poisson())
b = anova(null, full, test='Rao')

# Note: glm can also do the dummy coding for you:
c = glm(counts ~ mood, data=D, family=poisson())
```

Let's look at the results:

model	Chisq	df	p.value
chisq.test	6.3636	2	0.0415
glm LRT	6.3636	2	0.0415

Show R output

Note the strange `anova(..., test='Rao')` which merely states that p-values should be computed using the (Rao) score test (https://en.wikipedia.org/wiki/Score_test). We could also have jotted in `test='Chisq'` or `test='LRT'` which would have yielded approximate p-values. You may think that we're cheating here, sneaking in some sort of Chi-Square model post-hoc. However, `anova` only specifies how p-values are calculated whereas all the log-linear modeling happened in `glm` .

By the way, if there are only two counts and a large sample size (N > 100), this model begins to approximate the **binomial test**, `binom.test`, to a reasonable degree. But this sample size is larger than most use cases, so I won't raise to a rule-of-thumb and won't dig deeper into it here.

7.2 Contingency tables

7.2.1 Theory: As log-linear model

The theory here will be a bit more convoluted, and I mainly write it up so that you can get the *feeling* that it really is just a log-linear two-way ANOVA model. Let's get started…

For a two-way contingency table, the model of the count variable y is a modeled using the marginal proportions of a contingency table. Why this makes sense, is too involved to go into here, but see the relevant slides by Christoph Scheepers here (https://www.uni-tuebingen.de/fileadmin/Uni_Tuebingen/SFB/SFB_833/A_Bereich/A1/Christoph_Scheepers_-_Statistikworkshop.pdf) for an excellent exposition. The model is composed of a lot of counts and the regression coefficients A_i and B_i:

$$y_i = N \cdot x_i(A_i/N) \cdot z_j(B_j/N) \cdot x_{ij}/((A_i x_i)/(B_j z_j)/N)$$

What a mess!!! Here, i is the row index, j is the column index, $x_{something}$ is the sum of that row and/or column, $N = sum(y)$. Remember that y is a count variable, so N is just the total count.

We can simplify the notation by defining the *proportions*: $\alpha_i = x_i(A_i/N)$, $\beta_i = x_j(B_i/N)$ and $\alpha_i\beta_j = x_{ij}/(A_i x_i)/(B_j z_j)/N$. Let's write the model again:

$$y_i = N \cdot \alpha_i \cdot \beta_j \cdot \alpha_i\beta_j$$

Ah, much prettier. However, there is still lot's of multiplication which makes it hard to get an intuition about how the actual numbers interact. We can make it much more intelligible when we remember that $log(A \cdot B) = log(A) + log(B)$. Doing logarithms on both sides, we get:

$$log(y_i) = log(N) + log(\alpha_i) + log(\beta_j) + log(\alpha_i\beta_j)$$

Snuggly! Now we can get a better grasp on how the regression coefficients (which are proportions) independently contribute to y. This is why logarithms are so nice for proportions. Note that this is just the two-way ANOVA model with some logarithms added, so we are back to our good old linear models - only the interpretation of the regression coefficients have changed! And we cannot use `lm` anymore in R.

7.2.2 Example data

Here we need some long data and we need it in table format for `chisq.test`:

```
# Contingency data in long format for linear model
D = data.frame(mood = c('happy', 'happy', 'meh', 'meh', 'sad', 'sad'),
               sex = c('male', 'female', 'male', 'female', 'male', 'female'),
               Freq = c(100, 70, 30, 32, 110, 120))

# ... and as table for chisq.test
D_table = D %>%
  spread(key=mood, value=Freq) %>%  # Mood to columns
  select(-sex) %>%  # Remove sex column
  as.matrix()

# Dummy coding of D for linear model (skipping mood=="sad" and gender=="female")
# We could also use model.matrix(D$Freq~D$mood*D$sex)
D$mood_happy = ifelse(D$mood == 'happy', 1, 0)
D$mood_meh = ifelse(D$mood == 'meh', 1, 0)
D$sex_male = ifelse(D$sex == 'male', 1, 0)
```

mood	sex	Freq	mood_happy	mood_meh	sex_male
happy	male	100	1	0	1
happy	female	70	1	0	0
meh	male	30	0	1	1
meh	female	32	0	1	0

mood	sex	Freq	mood_happy	mood_meh	sex_male
sad	male	110	0	0	1
sad	female	120	0	0	0

7.2.3 R code: Chi-square test

Now let's show the equivalence between a chi-square model and a log-linear model. This is very similar to our two-way ANOVA above:

```
# Built-in chi-square. It requires matrix format.
a = chisq.test(D_table)

# Using glm to do a log-linear model, we get identical results when testing the interaction term:
full = glm(Freq ~ 1 + mood_happy + mood_meh + sex_male + mood_happy*sex_male + mood_meh*sex_male, data=D, family=poisson
())
null = glm(Freq ~ 1 + mood_happy + mood_meh + sex_male, data=D, family=poisson())
b = anova(null, full, test='Rao')  # Could also use test='LRT' or test='Chisq'

# Note: let glm do the dummy coding for you
full = glm(Freq ~ mood * sex, family=poisson(), data=D)
c = anova(full, test='Rao')

# Note: even simpler syntax using MASS:Loglm ("log-linear model")
d = MASS::loglm(Freq ~ mood + sex, D)
```

model	Chisq	df	p.value
chisq.test	5.0999	2	0.0781
glm	5.0999	2	0.0781
loglm	5.0999	2	0.0781

Show R output

If you unfold the raw R output, I've included `summary(full)` so that you can see the raw regression coefficients. Being a log-linear model, these are the *percentage increase*in y over and above the intercept if that category obtains.

8 Sources and further equivalences

Here are links to other sources who have exposed bits and pieces of this puzzle, including many further equivalences not covered here:

- My original exposition of the idea (https://stats.stackexchange.com/questions/303269/common-statistical-tests-as-linear-models) at StackOverflow
- An earlier question by me (https://stats.stackexchange.com/questions/210529/are-parametric-tests-on-rank-transformed-data-equivalent-to-non-parametric-test?noredirect=1#comment399981_210529) about non-parametric tests and a helpful answer.
- This question and replies (https://stats.stackexchange.com/questions/59047/how-are-regression-the-t-test-and-the-anova-all-versions-of-the-general-linear) on t-tests and ANOVA at StackOverflow
- These slides by Christoph Scheepers (https://www.uni-tuebingen.de/fileadmin/Uni_Tuebingen/SFB/SFB_833/A_Bereich/A1/Christoph_Scheepers_-_Statistikworkshop.pdf) on Chi-Square as log-linear models.
- This notebook by Philip M. Alday (https://rpubs.com/palday/glm-test) on Chi-square, binomial, multinomial, and poisson tests as log-linear and logistic models. These "equivalences" are less exact than what I presented above, and were therefore not included here. They are still great for a conceptual understanding of these tests, though!
- This article by Kristoffer Magnusson (https://rpsychologist.com/r-guide-longitudinal-lme-lmer) on RM-ANOVA and growth models using `lmer` mixed models.
- This post by Thom Baguley (https://seriousstats.wordpress.com/2012/02/14/friedman/) on the Friedman test. That post was actually the one that inititated my exploration of linear equivalences to non-parametric tests which ultimately pushed me over the edge to write up the present article.

9 Teaching materials and a course outline

Most advanced stats books (and some intro-books) take the “everything is GLMM” approach as well. However, the “linear model” part often stay at the conceptual level. I wanted to make linear models the *tool* in a really concise way. Luckily, more beginnier-friendly materials have emerged lately:

- Russ Poldracks open-source book “Statistical Thinking for the 21st century” (start at chapter 5 on modeling (http://statsthinking21.org/fitting-models-to-data.html))
- Jeff Rouder’s course notes (https://jeffrouder.blogspot.com/2019/03/teaching-undergrad-stats-without-p-f-or.html), introducing model comparison using just R^2 and BIC. It avoids all the jargon on p-values, F-values, etc. The full materials and slides are available here (https://drive.google.com/drive/folders/1CiJK--bAuO0F-ug3B5I3FvmsCdpPGZ03).

Here are my own thoughts on what I’d do. I’ve done parts of this with great success already, but not the whole lot since I’m not assigned to do a full course yet.

I would spend 50% of the time on linear modeling of data (bullet 1 below) since this contains 70% of what students need to know. The rest of the course is just fleshing out what happens if you have one group, two groups, etc.

Note that whereas the understanding of sampling and hypothesis testing is usually the first focus of mainstream stats courses, it is saved for later here to make way for modeling.

1. **Fundamentals of regression:**
    1. Recall from high-school: $y = a \cdot x + b$, and getting a really good intuition about slopes and intercepts. Understanding that this can be written using all variable names, e.g., `money = profit * time + starting_money` or $y = \beta_1 x + \beta_2 * 1$ or, suppressing the coefficients, as `y ~ x + 1`. If the audience is receptive, convey the idea of these models as a solution to differential equations (https://magesblog.com/post/modelling-change), specifying how y *changes* with x.
    2. Extend to a few multiple regression as models. Make sure to include plenty of real-life examples and exercises at this point to make all of this really intuitive. Marvel at how briefly these models allow us to represent large datasets.

3. Introduce the idea of rank-transforming non-metric data and try it out.
4. Teach the three assumptions: independence of data points, normality of residuals, and homoscedasticity.
5. Confidence/credible intervals on the parameters. Stress that the Maximum-Likelihood estimate is extremely unlikely, so intervals are more important.
6. Briefly introduce R^2 for the simple regression models above. Mention in passing that this is called the Pearson and Spearman correlation coefficients.

2. **Special case #1: One or two means (t-tests, Wilcoxon, Mann-Whitney):**
    1. *One mean:** When there is only one x-value, the regression model simplifies to $y = b$. If y is non-metric, you can rank-transform it. Apply the assumptions (homoscedasticity doesn't apply since there is only one x). Mention in passing that these intercept-only models are called one-sample t-test and Wilcoxon Signed Rank test respectively.
    2. **Two means:** If we put two variables 1 apart on the x-axis, the difference between the means is the slope. Great! It is accessible to our swizz army knife called linear modeling. Apply the assumption checks to see that homoscedasticity reduces to equal variance between groups. This is called an independent t-test. Do a few worked examples and exercises, maybe adding Welch's test, and do the rank-transformed version, called Mann-Whitney U.
    3. *Paired samples:* Violates the independence assumption. After computing pairwise differences, this is equivalent to 2.1 (one intercept), though it is called the paired t-test and Wilcoxon's matched pairs.
3. **Special case #2: Three or more means (ANOVAs)**
    1. *Dummy coding of categories:* How one regression coefficient for each level of a factor models an intercept for each level when multiplied by a binary indicator. This is just extending what we did in 2.1. to make this data accessible to linear modeling.
    2. *Means of one variable:* One-way ANOVA.
    3. *Means of two variables:* Two-way ANOVA.
4. **Special case #3: Three or more proportions (Chi-Square)**

1. *Logarithmic transformation:* Making multiplicative models linear using logarithms, thus modeling proportions. See this excellent introduction (https://www.uni-tuebingen.de/fileadmin/Uni_Tuebingen/SFB/SFB_833/A_Bereich/A1/Christoph_Scheepers_-_Statistikworkshop.pdf) to the equivalence of log-linear models and Chi-Square tests as models of proportions. Also needs to introduce (log-)odds ratios. When the multiplicative model is made summative using logarithms, we just add the dummy-coding trick from 3.1, and see that the models are identical to the ANOVA models in 3.2 and 3.3, only the interpretation of the coefficients have changed.
2. *Proportions of one variable:* Goodness of fit.
3. *Proportions of two variables:* Contingency tables.

5. **Hypothesis testing:** Hypothesis testing is the act of choosing between a full model and one where a parameter is set to zero (effectively excluded from the model) instead of being estimated. For example, when set one of the two means in the t-test to be zero, we study how well the remaining mean explains all the data from both groups. If it does a good job, we prefer this model over the two-mean model because it is simpler. So hypothesis testing is just comparing linear models to make more qualitative statements than the truly quantitative statements which were covered in bullets 1-4 above. Therefore, hypothesis testing is less interesting and is mostly covered as an introduction to the general literature. Mention P-values (and misconceptions about them), Bayes Factors, and cross-validation.

10 Limitations

I have made a few simplifications for clarity:

1. I have not covered assumptions in the examples. This will be another post! But all assumptions of all tests come down to the usual three: a) independence of data points, b) normally distributed residuals, and c) homoscedasticity.
2. I assume that all null hypotheses are the absence of an effect, but everything works the same for non-zero null hypotheses.

3. I have not discussed inference. I am only including p-values in the comparisons as a crude way to show the equivalences between the underlying models since people care about p-values. Parameter estimates will show the same equivalence. How to do *inference* is another matter. Personally, I'm a Bayesian, but going Bayesian here would render it less accessible to the wider audience. Also, doing robust models (https://en.wikipedia.org/wiki/Robust_statistics) would be preferable, but fail to show the equivalence.
4. Several named tests are still missing from the list and may be added at a later time. This includes the Sign test (require large N to be reasonably approximated by a linear model), Friedman as RM-ANOVA on `rank(y)`, McNemar, and Binomial/Multinomial. See stuff on these in the section on links to further equivalences. If you think that they should be included here, feel free to submit "solutions" to the github repo (https://github.com/lindeloev/tests-as-linear/) of this doc!